```
**********************************************************
    3097 Fri Jul 22 06:35:21 2016
new/hotspot/src/os/solaris/vm/jvm_solaris.h
**********************************************************
```

```
/*
 * Copyright (c) 1998, 2016, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 1998, 2015, Oracle and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License version 2 only, as
 * published by the Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Please contact Oracle, 500 Oracle Parkway, Redwood Shores, CA 94065 USA
 * or visit www.oracle.com if you need additional information or have any
 * questions.
 *
 */

#ifndef OS_SOLARIS_VM_JVM_SOLARIS_H
#define OS_SOLARIS_VM_JVM_SOLARIS_H

/*
// HotSpot integration note:
//
// This is derived from the JDK classic file:
// "$JDK/src/solaris/javavm/export/jvm_md.h":15 (ver. 1.10 98/04/22)
// All local includes have been commented out.
*/

#ifndef JVM_MD_H
#define JVM_MD_H

/*
 * This file is currently collecting system-specific dregs for the
 * JNI conversion, which should be sorted out later.
 */

#define __USE_LEGACY_PROTOTYPES__
#include <dirent.h>             /* For DIR */
#undef __USE_LEGACY_PROTOTYPES__
#include <sys/param.h>          /* For MAXPATHLEN */
#include <sys/socket.h>         /* For socklen_t */
#include <unistd.h>             /* For F_OK, R_OK, W_OK */
#include <sys/int_types.h>      /* for intptr_t types (64 Bit cleanliness) */

#define JNI_ONLOAD_SYMBOLS      {"JNI_OnLoad"}
#define JNI_ONUNLOAD_SYMBOLS    {"JNI_OnUnload"}
#define JVM_ONLOAD_SYMBOLS      {"JVM_OnLoad"}
#define AGENT_ONLOAD_SYMBOLS    {"Agent_OnLoad"}
#define AGENT_ONUNLOAD_SYMBOLS  {"Agent_OnUnload"}
#define AGENT_ONATTACH_SYMBOLS  {"Agent_OnAttach"}

#define JNI_LIB_PREFIX "lib"
#define JNI_LIB_SUFFIX ".so"
```



```
#define JVM_MAXPATHLEN MAXPATHLEN

#define JVM_R_OK    R_OK
#define JVM_W_OK    W_OK
#define JVM_X_OK    X_OK
#define JVM_F_OK    F_OK

/*
 * File I/O
 */

#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <errno.h>

/* O Flags */

#define JVM_O_RDONLY     O_RDONLY
#define JVM_O_WRONLY     O_WRONLY
#define JVM_O_RDWR       O_RDWR
#define JVM_O_O_APPEND   O_APPEND
#define JVM_O_EXCL       O_EXCL
#define JVM_O_CREAT      O_CREAT

/* Signal definitions */

#define BREAK_SIGNAL     SIGQUIT           /* Thread dumping support.    */
#define ASYNC_SIGNAL     SIGUSR2           /* Watcher & async err support. */
#define SHUTDOWN1_SIGNAL SIGHUP            /* Shutdown Hooks support.    */
#define SHUTDOWN2_SIGNAL SIGINT
#define SHUTDOWN3_SIGNAL SIGTERM

/* With 1.4.1 libjsig added versioning: used in os_solaris.cpp and jsig.c */
#define JSIG_VERSION_1_4_1   0x30140100

#endif /* JVM_MD_H */

#endif // OS_SOLARIS_VM_JVM_SOLARIS_H
```

****
39527 Fri Jul 22 06:35:21 2016
new/hotspot/src/os/solaris/vm/perfMemory_solaris.cpp
****
_____***unchanged_portion_omitted_***

```
298 // Open the directory of the given path and validate it.
299 // Return a DIR * of the open directory.
300 //
301 static DIR *open_directory_secure(const char* dirname) {
302   // Open the directory using open() so that it can be verified
303   // to be secure by calling is_dirfd_secure(), opendir() and then check
304   // to see if they are the same file system object.  This method does not
305   // introduce a window of opportunity for the directory to be attacked that
306   // calling opendir() and is_directory_secure() does.
307   int result;
308   DIR *dirp = NULL;
309   RESTARTABLE(::open(dirname, O_RDONLY|O_NOFOLLOW), result);
310   if (result == OS_ERR) {
311     // Directory doesn't exist or is a symlink, so there is nothing to cleanup.
312     if (PrintMiscellaneous && Verbose) {
313       if (errno == ELOOP) {
314         warning("directory %s is a symlink and is not secure\n", dirname);
315       } else {
316         warning("could not open directory %s: %s\n", dirname, os::strerror(errno
317       }
318     }
319     return dirp;
320   }
321   int fd = result;

323   // Determine if the open directory is secure.
324   if (!is_dirfd_secure(fd)) {
325     // The directory is not a secure directory.
326     os::close(fd);
327     return dirp;
328   }

330   // Open the directory.
331   dirp = ::opendir(dirname);
332   if (dirp == NULL) {
333     // The directory doesn't exist, close fd and return.
334     os::close(fd);
335     return dirp;
336   }

338   // Check to make sure fd and dirp are referencing the same file system object.
339   if (!is_same_fsobject(fd, dirp->d_fd)) {
339   if (!is_same_fsobject(fd, dirp->dd_fd)) {
340     // The directory is not secure.
341     os::close(fd);
342     os::closedir(dirp);
343     dirp = NULL;
344     return dirp;
345   }

347   // Close initial open now that we know directory is secure
348   os::close(fd);

350   return dirp;
351 }

353 // NOTE: The code below uses fchdir(), open() and unlink() because
354 // fdopendir(), openat() and unlinkat() are not supported on all
355 // versions.  Once the support for fdopendir(), openat() and unlinkat()
```



```
356 // is available on all supported versions the code can be changed
357 // to use these functions.

359 // Open the directory of the given path, validate it and set the
360 // current working directory to it.
361 // Return a DIR * of the open directory and the saved cwd fd.
362 //
363 static DIR *open_directory_secure_cwd(const char* dirname, int *saved_cwd_fd) {

365   // Open the directory.
366   DIR* dirp = open_directory_secure(dirname);
367   if (dirp == NULL) {
368     // Directory doesn't exist or is insecure, so there is nothing to cleanup.
369     return dirp;
370   }
371   int fd = dirp->d_fd;
371   int fd = dirp->dd_fd;

373   // Open a fd to the cwd and save it off.
374   int result;
375   RESTARTABLE(::open(".", O_RDONLY), result);
376   if (result == OS_ERR) {
377     *saved_cwd_fd = -1;
378   } else {
379     *saved_cwd_fd = result;
380   }

382   // Set the current directory to dirname by using the fd of the directory and
383   // handle errors, otherwise shared memory files will be created in cwd.
384   result = fchdir(fd);
385   if (result == OS_ERR) {
386     if (PrintMiscellaneous && Verbose) {
387       warning("could not change to directory %s", dirname);
388     }
389     if (*saved_cwd_fd != -1) {
390       ::close(*saved_cwd_fd);
391       *saved_cwd_fd = -1;
392     }
393     // Close the directory.
394     os::closedir(dirp);
395     return NULL;
396   } else {
397     return dirp;
398   }
399 }
```

_____***unchanged_portion_omitted_***

****
8891 Fri Jul 22 06:35:21 2016
new/hotspot/src/share/vm/utilities/globalDefinitions_sparcWorks.hpp
****

```
/*
 * Copyright (c) 1997, 2016, Oracle and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License version 2 only, as
 * published by the Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Please contact Oracle, 500 Oracle Parkway, Redwood Shores, CA 94065 USA
 * or visit www.oracle.com if you need additional information or have any
 * questions.
 *
 */

#ifndef SHARE_VM_UTILITIES_GLOBALDEFINITIONS_SPARCWORKS_HPP
#define SHARE_VM_UTILITIES_GLOBALDEFINITIONS_SPARCWORKS_HPP

#include "prims/jni.h"

// This file holds compiler-dependent includes,
// globally used constants & types, class (forward)
// declarations and a few frequently used utility functions.


# include <ctype.h>
#define __USE_LEGACY_PROTOTYPES__
# include <dirent.h>
#undef __USE_LEGACY_PROTOTYPES__
# include <string.h>
# include <strings.h>     // for bsd'isms
# include <stdarg.h>
# include <stddef.h>      // for offsetof
# include <stdio.h>
# include <stdlib.h>
# include <wchar.h>
# include <stdarg.h>
#ifdef SOLARIS
# include <ieeefp.h>
#endif
# include <math.h>
# include <time.h>
# include <fcntl.h>
# include <dlfcn.h>
# include <pthread.h>
#ifdef SOLARIS
# include <thread.h>
#endif
# include <limits.h>
# include <errno.h>
#ifdef SOLARIS
# include <sys/trap.h>
# include <sys/regset.h>
```



```
# include <sys/procset.h>
# include <ucontext.h>
# include <setjmp.h>
#endif
# ifdef SOLARIS_MUTATOR_LIBTHREAD
# include <sys/procfs.h>
# endif

#include <inttypes.h>

// Solaris 8 doesn't provide definitions of these
#ifdef SOLARIS
#ifndef PRIdPTR
#if defined(_LP64)
#define PRIdPTR                 "ld"
#define PRIuPTR                 "lu"
#define PRIxPTR                 "lx"
#else
#define PRIdPTR                 "d"
#define PRIuPTR                 "u"
#define PRIxPTR                 "x"
#endif
#endif
#endif

#ifdef LINUX
# include <signal.h>
# include <ucontext.h>
# include <sys/time.h>
#endif


// 4810578: varargs unsafe on 32-bit integer/64-bit pointer architectures
// When __cplusplus is defined, NULL is defined as 0 (32-bit constant) in
// system header files.  On 32-bit architectures, there is no problem.
// On 64-bit architectures, defining NULL as a 32-bit constant can cause
// problems with varargs functions: C++ integral promotion rules say for
// varargs, we pass the argument 0 as an int.  So, if NULL was passed to a
// varargs function it will remain 32-bits.  Depending on the calling
// convention of the machine, if the argument is passed on the stack then
// only 32-bits of the "NULL" pointer may be initialized to zero.  The
// other 32-bits will be garbage.  If the varargs function is expecting a
// pointer when it extracts the argument, then we have a problem.
//
// Solution: For 64-bit architectures, redefine NULL as 64-bit constant 0.
//
// Note: this fix doesn't work well on Linux because NULL will be overwritten
// whenever a system header file is included. Linux handles NULL correctly
// through a special type '__null'.
#ifdef SOLARIS
#ifdef _LP64
#undef NULL
#define NULL 0L
#else
#ifndef NULL
#define NULL 0
#endif
#endif
#endif

// NULL vs NULL_WORD:
// On Linux NULL is defined as a special type '__null'. Assigning __null to
// integer variable will cause gcc warning. Use NULL_WORD in places where a
// pointer is stored as integer value. On some platforms, sizeof(intptr_t) >
// sizeof(void*), so here we want something which is integer type, but has the
// same size as a pointer.
```

```
126 #ifdef LINUX
127   #ifdef _LP64
128     #define NULL_WORD  0L
129   #else
130     // Cast 0 to intptr_t rather than int32_t since they are not the same type
131     // on some platforms.
132     #define NULL_WORD  ((intptr_t)0)
133   #endif
134 #else
135   #define NULL_WORD  NULL
136 #endif

138 #ifndef LINUX
139 // Compiler-specific primitive types
140 typedef unsigned short     uint16_t;
141 #ifndef _UINT32_T
142 #define _UINT32_T
143 typedef unsigned int       uint32_t;
144 #endif
145 #if !defined(_SYS_INT_TYPES_H)
146 #ifndef _UINT64_T
147 #define _UINT64_T
148 typedef unsigned long long uint64_t;
149 #endif
150 // %%%% how to access definition of intptr_t portably in 5.5 onward?
151 typedef int                     intptr_t;
152 typedef unsigned int            uintptr_t;
153 // If this gets an error, figure out a symbol XXX that implies the
154 // prior definition of intptr_t, and add "&& !defined(XXX)" above.
155 #endif
156 #endif

158 // On solaris 8, UINTPTR_MAX is defined as empty.
159 // Everywhere else it's an actual value.
160 #if UINTPTR_MAX - 1 == -1
161 #undef UINTPTR_MAX
162 #ifdef _LP64
163 #define UINTPTR_MAX UINT64_MAX
164 #else
165 #define UINTPTR_MAX UINT32_MAX
166 #endif /* ifdef _LP64 */
167 #endif

169 // Additional Java basic types

171 typedef unsigned char      jubyte;
172 typedef unsigned short     jushort;
173 typedef unsigned int       juint;
174 typedef unsigned long long julong;


177 //--------------------------------------------------------------------------------
178 // Constant for jlong (specifying a long long constant is C++ compiler specific)

180 // Build a 64bit integer constant
181 #define CONST64(x)  (x ## LL)
182 #define UCONST64(x) (x ## ULL)

184 const jlong min_jlong = CONST64(0x8000000000000000);
185 const jlong max_jlong = CONST64(0x7fffffffffffffff);

187 #ifdef SOLARIS
188 //--------------------------------------------------------------------------------
189 // ANSI C++ fixes
190 // NOTE:In the ANSI committee's continuing attempt to make each version
191 // of C++ incompatible with the previous version, you can no longer cast
```



```
192 // pointers to functions without specifying linkage unless you want to get
193 // warnings.
194 //
195 // This also means that pointers to functions can no longer be "hidden"
196 // in opaque types like void * because at the invokation point warnings
197 // will be generated. While this makes perfect sense from a type safety
198 // point of view it causes a lot of warnings on old code using C header
199 // files. Here are some typedefs to make the job of silencing warnings
200 // a bit easier.
201 //
202 // The final kick in the teeth is that you can only have extern "C" linkage
203 // specified at file scope. So these typedefs are here rather than in the
204 // .hpp for the class (os:Solaris usually) that needs them.

206 extern "C" {
207    typedef int (*int_fnP_thread_t_iP_uP_stack_tP_gregset_t)(thread_t, int*, unsi
208    typedef int (*int_fnP_thread_t_i_gregset_t)(thread_t, int, gregset_t);
209    typedef int (*int_fnP_thread_t_i)(thread_t, int);
210    typedef int (*int_fnP_thread_t)(thread_t);

212    typedef int (*int_fnP_cond_tP_mutex_tP_timestruc_tP)(cond_t *cv, mutex_t *mx,
213    typedef int (*int_fnP_cond_tP_mutex_tP)(cond_t *cv, mutex_t *mx);

215    // typedef for missing API in libc
216    typedef int (*int_fnP_mutex_tP_i_vP)(mutex_t *, int, void *);
217    typedef int (*int_fnP_mutex_tP)(mutex_t *);
218    typedef int (*int_fnP_cond_tP_i_vP)(cond_t *cv, int scope, void *arg);
219    typedef int (*int_fnP_cond_tP)(cond_t *cv);
220 };
```

_____***unchanged_portion_omitted_***